

Unification

Mattox Beckman

beckman@iit.edu

Illinois Institute of Technology

Unification is a third major topic that will appear many times in this course. It is used in languages such as OCaml and Prolog, and also in theoretical discussions.

- Be able to describe the problem of unification.
- Be able to solve a unification problem.
- Know how to use unification to implement pattern matching.
- Know how to use unification to check types of functions.

Terms Have *name* and *arity*

- The name will be in western alphabet
- Arity = “number of arguments” — may be zero
- Examples: x , z , $f(x, y)$, $x(y, f, z)$

Variables Written using Greek alphabet, may be subscripted

- Represent a target for substitution
- Examples: α , β_{12} , γ_7

Substitutions Mappings from Variables to Terms

- Examples: $\sigma = \{\alpha \mapsto f(x, \beta), \beta \mapsto y\}$
- Substitutions are *applied*: $\sigma(g(\beta)) \rightarrow g(y)$

Note: arguments to terms may have non-zero arity, or may be variables.

- Given terms s and t , try to find a substitution σ such that $\sigma(s) = \sigma(t)$.
- If such a substitution exists, it is said that s and t unify.
- A *unification problem* is a set of equations $S = \{s_1 =? t_1, s_2 =? t_2, \dots\}$.
- A unification problem $S = \{x_1 =? t_1, x_2 =? t_2, \dots\}$ is in *solved form* if
 - the terms x_i are distinct variables
 - none of them occur in t_i .

Our approach: given a unification problem S , we want to find the most general unifier σ that solves it. We will do this by transforming the equations.

Start with a unification problem $S = \{s_1 =? t_1, s_2 =? t_2, \dots\}$ and apply the following transformations as necessary:

Delete A trivial equation $t =? t$ can be deleted.

Decompose An equation $f(\overline{t_n}) =? f(\overline{u_n})$ can be replaced by the set $\{t_1 =? u_1, \dots, t_n =? u_n\}$

Orient An equation $t =? x$ can be replaced by $x =? t$ if x is a variable and t is not.

Eliminate an equation $x =? t$ can be used to substitute all occurrences of x in the remainder of S .

(Stolen from “Term Rewriting and All That”)

$$\{\alpha =? f(x), g(\alpha, \alpha) =? g(\alpha, \beta)\}$$

(Stolen from “Term Rewriting and All That”)

$$\{\alpha =? f(x), g(\alpha, \alpha) =? g(\alpha, \beta)\}$$

We can use the Eliminate method, replace α with $f(x)$ on the right sides of the equations.

(Stolen from “Term Rewriting and All That”)

$$\{\alpha =? f(x), g(\alpha, \alpha) =? g(\alpha, \beta)\}$$

We can use the Eliminate method, replace α with $f(x)$ on the right sides of the equations.

$$\{\alpha =? f(x), g(f(x), f(x)) =? g(f(x), \beta)\}$$

We can use the Decompose method, and get rid of the g functions.

(Stolen from “Term Rewriting and All That”)

$$\{\alpha =? f(x), g(\alpha, \alpha) =? g(\alpha, \beta)\}$$

We can use the Eliminate method, replace α with $f(x)$ on the right sides of the equations.

$$\{\alpha =? f(x), g(f(x), f(x)) =? g(f(x), \beta)\}$$

We can use the Decompose method, and get rid of the g functions.

$$\{\alpha =? f(x), f(x) =? f(x), f(x) =? \beta\}$$

We can delete the $f(x) = f(x)$ equation.

(Stolen from “Term Rewriting and All That”)

$$\{\alpha =? f(x), g(\alpha, \alpha) =? g(\alpha, \beta)\}$$

We can use the Eliminate method, replace α with $f(x)$ on the right sides of the equations.

$$\{\alpha =? f(x), g(f(x), f(x)) =? g(f(x), \beta)\}$$

We can use the Decompose method, and get rid of the g functions.

$$\{\alpha =? f(x), f(x) =? f(x), f(x) =? \beta\}$$

We can delete the $f(x) = f(x)$ equation.

$$\{\alpha =? f(x), f(x) =? \beta\}$$

Now we can reorient to make the variables show up on the left side.

(Stolen from “Term Rewriting and All That”)

$$\{\alpha =? f(x), g(\alpha, \alpha) =? g(\alpha, \beta)\}$$

We can use the Eliminate method, replace α with $f(x)$ on the right sides of the equations.

$$\{\alpha =? f(x), g(f(x), f(x)) =? g(f(x), \beta)\}$$

We can use the Decompose method, and get rid of the g functions.

$$\{\alpha =? f(x), f(x) =? f(x), f(x) =? \beta\}$$

We can delete the $f(x) = f(x)$ equation.

$$\{\alpha =? f(x), f(x) =? \beta\}$$

Now we can reorient to make the variables show up on the left side.

$$\{\alpha =? f(x), \beta =? f(x)\}$$

Now we are done....

$$S = \{\alpha \mapsto f(x), \beta \mapsto f(x)\}$$

- Pattern Matching is one form of unification.

```
1 let lst = 3::4::5::[ ];;  
2 match lst with  
3   | [ ] -> ...  
4   | x::xs -> ....
```

- We want to unify lst with $[]$ or $x :: xs$.
 - Let $S_1 = \{[] =^? \alpha_{lst}, \alpha_{lst} =^? 3::4::5::[]\}$
 - Let $S_2 = \{\alpha_x :: \alpha_{xs} =^? \alpha_{lst}, \alpha_{lst} =^? 3::4::5::[]\}$
- What is the solution?

- Let $S_1 = \{[] =^? \alpha_{lst}, \quad \alpha_{lst} =^? 3 :: 4 :: 5 :: []\}$
 - Substitution: $S_1 = \{[] =^? 3 :: 4 :: 5 :: []\}$
 - Fails to unify.
- Let $S_2 = \{\alpha_x :: \alpha_{xs} =^? \alpha_{lst}, \quad \alpha_{lst} =^? 3 :: 4 :: 5 :: []\}$
 - Substitution: $\{\alpha_x :: \alpha_{xs} =^? 3 :: 4 :: 5 :: []\}$
 - Decomposition: $\{\alpha_x =^? 3, \quad \alpha_{xs} =^? 4 :: 5 :: []\}$

- Your advisor wants you to take CS 440 and some theory class.
- Your mom wants you to take CS 536 and some languages class.
- Can both your advisor and your mom be happy?

This is a problem we can solve using unification:

- Let f be a “schedule function”, the first argument is a language class, the second argument is a theory class.
- $s = f(cs440, \beta)$ (where β is a theory class)
- $t = f(\alpha, cs536)$ (where α is a language class)
- Let $\sigma = \{\alpha \mapsto cs440, \beta \mapsto cs536\}$

Type checking is also a form of unification.

```
1 map : ('a -> 'b) -> 'a list -> 'b list
2 inc : int -> int
3 foo : int list
```

Will `map inc foo` work?

$$S = \{(\alpha \rightarrow \beta) =^? \text{int} \rightarrow \text{int}, \quad (\alpha \text{ list}) =^? \text{int list}\}$$

$$S = \{(\alpha \rightarrow \beta) =^? \text{int} \rightarrow \text{int}, \quad (\alpha \text{ list}) =^? \text{int list}\}$$

- Decompose: $\{\alpha =^? \text{int}, \quad \beta =^? \text{int}, \quad (\alpha \text{ list}) =^? \text{int list}\}$
- Substitute: $\{\alpha =^? \text{int}, \quad \beta =^? \text{int}, \quad (\text{int list}) =^? \text{int list}\}$
- Delete: $\{\alpha =^? \text{int}, \quad \beta =^? \text{int}\}$

The original type of `map` was $(\alpha \rightarrow \beta) \rightarrow \alpha \text{ list} \rightarrow \beta \text{ list}$

We can use our pattern to get the output type:

$$S(\beta \text{ list}) \Rightarrow \text{int list}$$

Here's an example that fails.

```
1 map : ('a -> 'b) -> 'a list -> 'b list  
2 inc : string -> int  
3 foo : int list
```

Will `map inc foo` work?

$$S = \{(\alpha \rightarrow \beta) =? \text{string} \rightarrow \text{int}, \quad (\alpha \text{ list}) =? \text{int list}\}$$

$$S = \{(\alpha \rightarrow \beta) =^? \text{string} \rightarrow \text{int}, \quad (\alpha \text{ list}) =^? \text{int list}\}$$

- Decompose: $\{\alpha =^? \text{string}, \quad \beta =^? \text{int}, \quad (\alpha \text{ list}) =^? \text{int list}\}$
- Substitute:
 $\{\alpha =^? \text{string}, \quad \beta =^? \text{int}, \quad (\text{string list}) =^? \text{int list}\}$
- Error: $(\text{string list}) \neq^? \text{int list}!$

Try to unify the following:

• $\{f(\alpha, y) = f(x, \beta)\}$

• $\{f(\alpha, y) = f(x, \alpha)\}$

• $\{f(\alpha, \beta) = \gamma, \quad \gamma = f(x, \delta), \quad \beta = g(y)\}$

• $\{f(\alpha, \beta) = \gamma, \quad \gamma = f(x, \delta)\}$

● $\{f(\alpha, y) = f(x, \beta)\}$

Let $\alpha \mapsto x, \quad \beta \mapsto y$

● $\{f(\alpha, y) = f(x, \alpha)\}$

This one cannot be unified.

● $\{f(\alpha, \beta) = \gamma, \quad \gamma = f(x, \delta), \quad \beta = g(y)\}$

Let $\{\beta, \delta \mapsto g(y), \quad \alpha \mapsto x, \quad \gamma \mapsto f(x, g(y))\}$

● $\{f(\alpha, \beta) = \gamma, \quad \gamma = f(x, \delta)\}$

Let $\{\beta \mapsto \delta, \quad \alpha \mapsto x, \quad \gamma \mapsto f(x, \beta)\}$

Unifies, but not completely solved.